# Sobol Tensor Trains for Global Sensitivity Analysis

Rafael Ballester-Ripoll[a,*], Enrique G. Paredes[a], Renato Pajarola[a]

[a] *Visualization and MultiMedia Lab, Department of Informatics, University of Zurich, Switzerland*

**Abstract**

Sobol indices are a widespread quantitative measure for variance-based global sensitivity analysis, but computing and utilizing them remains challenging for high-dimensional systems. We propose the tensor train decomposition (TT) as a unified framework for surrogate modeling and sensitivity analysis via Sobol indices. We first overview several strategies to build a TT surrogate using either an adaptive sampling strategy or a predefined set of samples. Our main contribution is the introduction of the Sobol TT, which compactly represents variance components for all possible joint variable interactions of any order. Our formulation allows efficient aggregation and subselection operations, and we are able to obtain related Sobol indices (closed, total, and superset indices) at negligible cost. Furthermore, we exploit an existing global optimization procedure within the TT framework for variable selection and model analysis tasks. We demonstrate our algorithms with two analytical models and a parallel computing simulation data set.

*Keywords:* Sensitivity analysis, surrogate modeling, Sobol indices, tensor trains, low-rank approximation

## 1. Introduction

A crucial task when analyzing computational models and physical simulations is assessing the influence of each input variable (and all combinations thereof) on the model's output. The quantitative study of such influences is known as *sensitivity analysis* (SA). When the variables are stochastic the propagation of their uncertainty towards the model output must also be taken into account. We focus on variance-based SA, often referred to as *analysis of variances* (ANOVA), and in particular the so-called *Sobol decomposition*. It approximates the parametrized model as a sum of simpler functions, each depending on only a subset of the original set of variables. The sensitivity to each variable

*Corresponding author

*Email addresses:* rballester@ifi.uzh.ch (Rafael Ballester-Ripoll), egparedes@ifi.uzh.ch (Enrique G. Paredes), pajarola@ifi.uzh.ch (Renato Pajarola)

is then reflected by the functions that depend on it, and can therefore be estimated as their relative contribution to the output's overall statistical variance. These relative variances have become a standard tool for global SA in the last few decades [1, 2, 3, 4, 5].

A popular method to compute such indices is via *Monte Carlo* (MC) integration estimators on a suitable set of samples within the variable space (the *sampling plan*). This was already outlined in the original paper by Sobol [6] and gained momentum thereafter. However, MC convergence is slow with respect to the number of samples drawn [5]. Structured sampling plans exist that improve convergence, e.g. *quasi-random sequences* (quasi-MC) or *Latin hypercube sampling* (LHS) [7]. If needed one may favor estimators for *total effect* indices, i.e. metrics (QOI) that aggregate quantities of interest (QOI) of diverse orders together. Unfortunately, a plan tailored to estimate a particular index, or set thereof, may be suboptimal for other indices. In practice, analysts often choose to restrict the Sobol decomposition to interactions of low-order (e.g. up to 2), and/or perform a prior dimensionality reduction in what is known as *screening* (e.g. freezing seemingly unimportant variables). Such simplifications greatly reduce the computational complexity, but pose a risk: they might fail to detect significant complex interactions between variables, and over-zealous reduction can harm subsequent processing steps in the analysis pipeline.

A complementary approach to direct MC estimation is building a *surrogate model*, also known as *response surface model* or *metamodel*, in an offline preliminary step. This includes, among others, Gaussian processes [8] or radial basis function interpolators [9]. The surrogate acts as an interpolator that is fast to evaluate and can approximate the true unknown model at arbitrary sampling points [10]. This strategy is attractive when sample acquisition is expensive or highly dynamic, especially if the analyst would like to estimate new indices on demand. Certain families of surrogates can produce Sobol indices in a more direct manner [5, 9], thus avoiding MC integration altogether. However, dealing with high-dimensional parametric systems, i.e. with a significant number $N$ of input variables, remains a major challenge. Even if the chosen surrogate scales well with the dimensionality [11], the sheer number of sensitivity indices is by definition exponential, as there are $2^N - 1$ variance components, out of which $\binom{N}{M}$ for any fixed order $1 \leq M \leq N$ may be chosen. For moderate or large values of $N$, general queries of the form *"retrieve the largest indices of any order"* or *"compute the total variance for interactions of order up to $k$"* quickly become computationally intractable.

To address these problems we propose a data structure that compactly stores *all* Sobol sensitivity indices in a compressed form. It is based on tensor decompositions, in particular the tensor train (TT) [12] method. The TT format is designed to avoid the curse of dimensionality and excels in high-dimensional approximation and compression in several fields ranging from physics and quantum chemistry to engineering and data mining. It can be built using various sampling/interpolation techniques, both when a sample acquisition plan is required and when the set of known samples is fixed beforehand.

While formulas to compute individual or aggregated sensitivity indices from

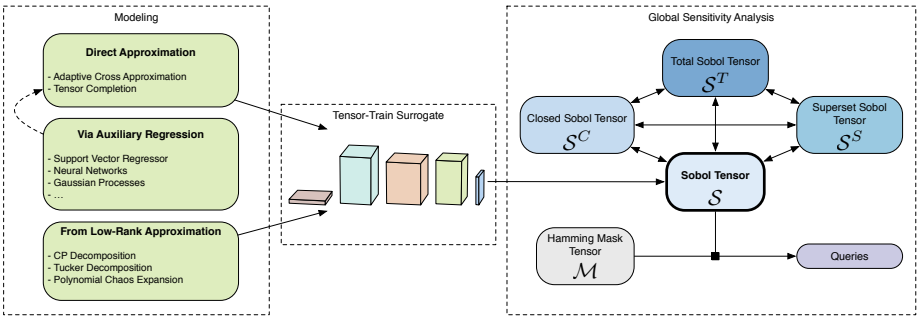



Figure 1: Pipeline for TT-based global sensitivity analysis: a model with $N$ input variables is approximated as an $N$-dimensional tensor, from which we extract a compact $2^N$ tensor $\mathcal{S}$ approximating all $2^N - 1$ variance components. This tensor can be then used for various aggregation, analysis and query/optimization tasks.

various low-rank surrogates have been already derived in the recent literature [13, 14, 11], our approach is the first to assemble the complete set of all indices in a unified and compact tensor format that can be manipulated and queried for statistics, model reduction, visual analytics, and more. See Fig. 1 for a summary diagram of our pipeline.

### *1.1. Contributions*

We introduce the Sobol tensor $\mathcal{S}$, an $N$-dimensional TT-compressed multiarray encoding all possible $2^N - 1$ variance components for global SA, and show its derivation from an arbitrary TT surrogate model. We further extract the related aggregated tensors $\mathcal{S}^S$, $\mathcal{S}^C$ and $\mathcal{S}^T$ containing the *superset*, *closed* and *total* indices, respectively. All these indices can be derived from each other via union/intersection operations that are translated to the tensor-compressed domain as simple matrix additions and subtractions. By combining these ideas with existing optimization algorithms for the TT format we are able to answer several computationally challenging types of global SA queries that often arise during variable selection and model interpretation.

### *1.2. Notation*

Multidimensional arrays, herein called *tensors*, have sizes denoted by $I_1, ..., I_N$ where $N$ is the number of dimensions. *Tensor ranks* generalize the matrix rank for $N > 2$ and reflect, in a sense, the complexity of a tensor; they use the symbols $R_n$. For simplicity we sometimes use $I := \max\{I_n\}_n$ and $R := \max\{R_n\}_n$. Vectors, matrices and tensors use bold lowercase, bold uppercase and calligraphic letters as in $\boldsymbol{x}$, $\mathbf{U}$ and $\mathcal{T}$ respectively. Their elements are denoted by square brackets with indices starting from 0, following NumPy convention (e.g. $\mathbf{U}[1, 0]$). Furthermore, we refer to the element-wise (i.e., Hadamard) product of two tensors as $\mathcal{A} \circ \mathcal{B}$, whereas the Kronecker product of matrices is written as $\mathbf{A} \otimes \mathbf{B}$. We use the Frobenius norm $\|\cdot\| := \|\cdot\|_2$ for tensors. Accuracy between

a groundtruth tensor $\mathcal{A}$ and an approximation $\mathcal{B}$ is measured with the *relative error*: $\epsilon_{\mathcal{A}}(\mathcal{B}) := \|\mathcal{A} - \mathcal{B}\|/\|\mathcal{A}\| \geq 0$.

We denote tuples of indices as $\boldsymbol{\alpha} \in \{0,1\}^N$ and $\boldsymbol{\alpha} \subseteq \{1,...,N\}$ interchangeably: a 0 (resp. 1) in the former notation means an index is absent (resp. present) in the latter. If a function $f : \mathbb{R}^4 \to \mathbb{R}$ only depends *effectively* on the two last variables, we may alternatively write $\boldsymbol{\alpha} = [0,0,1,1]$ or $\boldsymbol{\alpha} = \{3,4\}$, and $f_{\boldsymbol{\alpha}}(\boldsymbol{x}) \equiv f_{\boldsymbol{\alpha}}(\boldsymbol{x}_{\boldsymbol{\alpha}}) \equiv f_{3,4}(x_3, x_4)$ similarly to [2, 15]. Cardinality of a set of variables is denoted as $|\boldsymbol{\alpha}|$, and coincides with the *Hamming weight* (bit sum) of its binary representation. Last, we write tuple complements as $-\boldsymbol{\alpha} \equiv \{1,...,N\} \setminus \boldsymbol{\alpha}$.

## 2. Variance-Based Sensitivity Analysis

### *2.1. Sobol Decomposition*

Variance-based SA dates back to the early 20th century and comprises a set of related techniques for statistical analysis of multidimensional data, out of which the ANOVA is arguably the most widely known. The functional ANOVA decomposition [16], also known as high-dimensional model representation (HDMR) [17] or Sobol decomposition [6], writes any integrable multidimensional function over a rectangle $f : \Omega = \Omega_1 \times ... \times \Omega_N \subset \mathbb{R}^N \to \mathbb{R}$ as the following sum of subfunctions:

$$\begin{aligned} f(\boldsymbol{x}) = f_{\emptyset} + \sum_{i=1}^{N} f_i(x_i) + \sum_{1 \leq i < j \leq N} f_{ij}(x_i, x_j) + \dots \\ + \sum_{\substack{\boldsymbol{\alpha} \subset \{1,\dots,N\} \\ |\boldsymbol{\alpha}| = n}} f_{\boldsymbol{\alpha}}(x_{\boldsymbol{\alpha}}) + \dots + f_{1,\dots,N}(x_1, ..., X_N) \end{aligned} \tag{1}$$

where each $f_{\boldsymbol{\alpha}}$ only depends effectively on the indices contained in $\boldsymbol{\alpha}$. The $f_{\boldsymbol{\alpha}}$ are uniquely determined if orthogonality w.r.t. a separable measure $dF(\boldsymbol{x}) = dF_1(x_1) \cdots dF_N(x_N)$ is imposed:

$$\int_{\Omega} f_{\boldsymbol{\alpha}}(\boldsymbol{x}_{\boldsymbol{\alpha}}) f_{\boldsymbol{\beta}}(\boldsymbol{x}_{\boldsymbol{\beta}})\, dF(\boldsymbol{x}) = 0 \text{ for any } \boldsymbol{\alpha} \neq \boldsymbol{\beta} \tag{2}$$

Then, the explicit decomposition stems from iterative integrations and subtractions [6]:

$$\begin{aligned} f_\emptyset &= \int_\Omega f(\boldsymbol{x})\, dF(\boldsymbol{x}) \\ f_n(x_n) &= \int_{\Omega_{-n}} f(\boldsymbol{x})\, dF_{-n}(\boldsymbol{x}_{-n}) - f_\emptyset \\ f_{nm}(x_n, x_m) &= \int_{\Omega_{-nm}} f(\boldsymbol{x})\, dF_{-nm}(\boldsymbol{x}_{-nm}) \\ &\quad - f_n(x_n) - f_m(x_m) - f_\emptyset \\ &\quad \dots \\ f_{\boldsymbol{\alpha}}(\boldsymbol{x}_{\boldsymbol{\alpha}}) &= \int_{\Omega_{-\boldsymbol{\alpha}}} f(\boldsymbol{x})\, dF_{-\boldsymbol{\alpha}}(\boldsymbol{x}_{-\boldsymbol{\alpha}}) - \sum_{\boldsymbol{\beta}|\boldsymbol{\beta}\subset\boldsymbol{\alpha}} f_{\boldsymbol{\beta}}(\boldsymbol{x}_{\boldsymbol{\beta}}) \end{aligned} \tag{3}$$

Eqs. 1 to 3 are useful in the context of uncertainty quantification, namely when one has a model depending on $N$ independent random variables $x_1, ..., x_N$. Under this assumption, their joint *probability density function* (PDF) plays the role of our separable measure in $\Omega$ and the integrals are expectations of each subfunction, starting with $f_\emptyset = \mathrm{E}[f]$. Eq. 3 is then

$$f_{\boldsymbol{\alpha}}(\boldsymbol{x}_{\boldsymbol{\alpha}}) = (\mathrm{E}_{-\boldsymbol{\alpha}}[f])(\boldsymbol{x}_{\boldsymbol{\alpha}}) - \sum_{\boldsymbol{\beta}|\boldsymbol{\beta}\subset\boldsymbol{\alpha}} f_{\boldsymbol{\beta}}(\boldsymbol{x}_{\boldsymbol{\beta}}) \tag{4}$$

### *2.2. Variance Components*

The *unnormalized variance components* $D_{\boldsymbol{\alpha}}$ are defined as the variance contributed by each of the $f_{\boldsymbol{\alpha}}$, w.r.t. the measure $F$. Thus, the Sobol decomposition is a partition of the overall variance $D$:

$$D = \sum_{\boldsymbol{\alpha}} D_{\boldsymbol{\alpha}} = \mathrm{V}[f] = \mathrm{E}[f^2] - \mathrm{E}^2[f] = \int_\Omega f(\boldsymbol{x})^2 dF(\boldsymbol{x}) - f_\emptyset^2 \tag{5}$$

The *normalized variance components* (or simply *variance components*) in turn measure the relative variances w.r.t. the total model variance:

$$\begin{gathered} S : \mathcal{P}(\{1, ..., N\}) \setminus \emptyset \to [0, 1] \\ S_{\boldsymbol{\alpha}} := D_{\boldsymbol{\alpha}}/D \\ \textstyle\sum_{\boldsymbol{\alpha}} S_{\boldsymbol{\alpha}} = 1 \end{gathered} \tag{6}$$

These indices are an invaluable tool in many SA settings [18], for example in factor prioritization (reducing uncertainty), factor fixing (identifying non-influential variables), risk minimization, reliability engineering, etc. They are also helpful to select good dimension orderings that lead to more compact surrogate models (example 5.8 by Bigoni [17]; also considered in [14]). They are hyperedges of a hypergraph, since they encode $n$-ary relations within subsets of

$\{1, ..., N\}$. Alternatively they can be thought of in terms of set cardinalities, as the sum of all $S_{\boldsymbol{\alpha}}$ equals 1 (see e.g. [19] and [1], Sec. 1.2.15).

Several surrogate models lend themselves well to direct estimation of Sobol indices. Examples in the literature include PCE of bounded degree [2], low-rank sums of separable functions [20, 11], Gaussian processes [3], TT [14], spectral TT [21], etc. However, there are $2^N - 1$ possible QOI after excluding the trivial tuple $\boldsymbol{\alpha} = [0, ..., 0] \equiv \emptyset$. As $N$ grows, this magnitude poses challenges in both the computational and the model interpretation aspects.

### *2.3. Related Sobol Indices*

Sobol's method defines alternative indices by adding and/or subtracting together the standard $S_{\boldsymbol{\alpha}}$, effectively configuring a set algebra.

#### *2.3.1. Total Indices*

Denoted as $S^T_{\boldsymbol{\alpha}}$, they are also called *upper indices* [19]. They represent all joint indices that include any variable from $\boldsymbol{\alpha}$:

$$S^T_{\boldsymbol{\alpha}} := \sum_{\boldsymbol{\beta} | \boldsymbol{\alpha} \cap \boldsymbol{\beta} \neq \emptyset} S_{\boldsymbol{\beta}} \tag{7}$$

For example, in a 3-variable model we have $S^T_{1,2} = S_1 + S_2 + S_{1,2} + S_{1,3} + S_{2,3} + S_{1,2,3}$. If $|\boldsymbol{\alpha}| = 1$ we are encoding the total influence of a single variable also accounting for its higher-order interactions with all other variables. In this case the indices are called sometimes *total effects* [22], and have been used to identify and select the most relevant variables, for example by sorting $S^T_n$ and choosing the $k$ largest [23, 24]. However, this criterion may lead to overestimating variables that exhibit large overlapping variance contributions.

#### *2.3.2. Closed Indices*

Denoted as $S^C_{\boldsymbol{\alpha}}$, they are also called *first-order indices* [2] or *lower indices* [19]. They sum the variance contributions of all non-empty tuples contained in $\boldsymbol{\alpha}$:

$$S^C_{\boldsymbol{\alpha}} := \sum_{\boldsymbol{\beta} | \boldsymbol{\alpha} \supseteq \boldsymbol{\beta}} S_{\boldsymbol{\beta}} \tag{8}$$

For example, for 3 variables we have $S^C_{1,2} = S_1 + S_2 + S_{1,2}$. Also, for any single variable $n$ we have $S^C_n = S_n$. The closed indices can be written in terms of the total indices as $S^C_{\boldsymbol{\alpha}} = 1 - S^T_{-\boldsymbol{\alpha}}$.

#### *2.3.3. Superset Indices*

The $\mathcal{S}^S$ aggregate all indices that contain a tuple [25]:

$$S^S_{\boldsymbol{\alpha}} := \sum_{\boldsymbol{\beta} | \boldsymbol{\alpha} \subseteq \boldsymbol{\beta}} S_{\boldsymbol{\beta}} \tag{9}$$

For example, $S^S_{1,2} = S_{1,2} + S_{1,2,3}$.

## 3. Tensor Approximation

Decomposing multidimensional arrays (tensors) in terms of simpler, separable terms is a fruitful approach in compression, interpolation and metamodeling applications, and their fundamentals reach out to other important mathematical frameworks including principal component analysis, wavelet transforms, polynomial chaos, etc. We briefly introduce first CP and Tucker since they are arguably the two most popular tensor models, and we support them in Sec. 4 as optional intermediate surrogate models. We cover then the more recent TT decomposition, which is the keystone of all algorithms presented in this paper. The section concludes with related work on tensor-based surrogate modeling and SA.

### *3.1. CANDECOMP/PARAFAC*

The CP, also known as *canonical* or *Kruskal decomposition*, is the earliest and most straightforward extension of the singular value decomposition (SVD) for more than 2 dimensions [26]. It approximates a tensor $\mathcal{T}$ element-wise as follows:

$$\mathcal{T}[x_1, ..., x_N] \approx \sum_{r=1}^{R} \lambda_r \cdot \mathbf{U}^{(1)}[x_1, r] \cdot ... \cdot \mathbf{U}^{(N)}[x_N, r] \tag{10}$$

where $R$ is the *CP rank*. The $\mathbf{U}^{(n)}$ are known as *factor matrices* or simply *factors*. We can write Eq. 10 more compactly using double bracket notation as $\mathcal{T} \approx [[\boldsymbol{\lambda}; \mathbf{U}^{(1)}, ..., \mathbf{U}^{(N)}]]$. The $\boldsymbol{\lambda}$ can be optionally absorbed column-wise by the factors $\mathbf{U}^{(n)}$ and omitted in the notation. Unfortunately, the set of $N$-dimensional tensors of fixed CP-rank $R$ is not closed in $\mathbb{R}^N$, and finding the best rank-$R$ approximation of a given tensor is an ill-posed problem [27]. On the positive side, the CP format needs $O(INR)$ elements for storage, i.e. it is linear w.r.t. the number of dimensions.

### *3.2. Tucker*

The Tucker decomposition [28, 29] is also known as *higher-order SVD* (HOSVD), *N-mode PCA*, and *low multilinear rank approximation* (LMLRA). It extends CP by considering all interactions between its factor columns, weighted by an $N$-dimensional core $\mathcal{B}$ of size $R_1 \times ... \times R_N$:

$$\mathcal{T}[x_1, ..., x_N] \approx \sum_{\boldsymbol{r=1}}^{\boldsymbol{R}} \mathcal{B}[\boldsymbol{r}] \cdot \mathbf{U}^{(1)}[x_1, r_1] \cdot ... \cdot \mathbf{U}^{(N)}[x_N, r_N] \tag{11}$$

Approximating a tensor with the Tucker format is a stable procedure [29]. However, $O(R^N + INR)$ elements need to be stored, i.e. it still suffers from the curse of dimensionality. It is therefore mostly used for up to a handful of dimensions only. The Tucker model is related to *polynomial chaos expansions* (PCE), as we detail later in Sec. 4.5.3.

### 3.3. Tensor Train

The TT model (also known since the 1990s as *matrix product states* or *linear tensor network*) was recently rediscovered by Oseledets [12]. It aims to unite the advantages of both CP and Tucker, especially for high $N$. It uses a sequence of 3D cores, compactly written as $[[\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(n)}]]$. The reconstruction is a sequence of matrix products:

$$\mathcal{T}[x_1, \ldots, x_N] = \mathcal{T}^{(1)}[x_1] \cdot \ldots \cdot \mathcal{T}^{(N)}[x_N] \tag{12}$$

where $\mathcal{T}^{(n)}[x_n]$ is a shorthand for the $x_n$-th slice along mode 2, i.e. $\mathcal{T}^{(n)}[:, x_n, :]$. The core dimensions are $R_{n-1} \times I_n \times R_n$ for $n = 1, ..., N$, with $R_0 := R_N := 1$ by convention. The $R_n$ are called *TT ranks* and are bounded from above by CP's $R$ rank. Fig. 2 illustrates the structure and indexing of the core sequence.

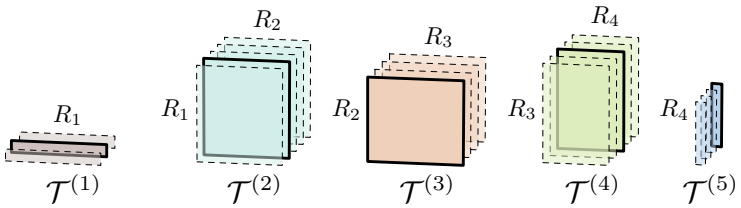



Figure 2: A 5D tensor train of size $3 \times 5 \times 4 \times 5 \times 4$. By multiplying the highlighted matrices together we obtain the element $\mathcal{T}^{(1)}[1] \cdot \mathcal{T}^{(2)}[1] \cdot \mathcal{T}^{(3)}[0] \cdot \mathcal{T}^{(4)}[2] \cdot \mathcal{T}^{(5)}[4] = \mathcal{T}[1, 1, 0, 2, 4]$.

In contrast to the Tucker model, the TT format needs $O(INR^2)$ elements and thus grows linearly w.r.t the number of dimensions $N$.

#### 3.3.1. Operations in the TT Format

Multiplication by a scalar and tensor-wise addition/product may be achieved by simple manipulations of the TT cores as shown in [12]; see also Appendix A for more in-depth details. Furthermore, thanks to the so-called *adaptive cross-approximation* (ACA) framework in the TT format [30, 31], these and many other operations can be accomplished in $O(INR^3)$ operations at most, i.e. devoid of the curse of dimensionality. These include arbitrary element-wise functions, differentiation, integration, convolution, and more [32, 33]. The ranks may grow as a result of such operations. It is crucial to keep them reasonably low at all stages of any computational pipeline, otherwise the benefits of tensor compression vanish. An error-bounded rounding algorithm called TT-round [12] exists to re-compress down any tensor when needed.

#### 3.3.2. Global Optimization

ACA has been successfully used to find the (approximately) maximal element in modulus of a TT tensor [14, 34], as it was empirically found that the subtensors accessed during cross-approximation very often contain such maximal elements. The algorithmic variant known as *rectangular maxvol* is a tool even more effective for this task [35] and is the one we use (as released in [36]).

### 3.4. Tensor Surrogates and Sensitivity Analysis

Tensor decompositions make for attractive surrogates owing to their natural multidimensionality and fast decompression. Several examples [37, 38, 14] target solutions of multiparametric partial differential equations (PDEs). Konakli and Sudret [11] propose an interpolator via sums of separable PCE-based functions (low-rank approximations, LRA) and showed how to extract Sobol indices out of them. This is related to the CP decomposition, with the main difference that their factors are continuous and are sought within the subspace spanned by a few leading orthogonal polynomials. Vervliet et al. [39] demonstrate CP-based tensor completion and visualization for the melting point of an alloy, depending on the concentration of its 10 different constituent materials. Ballester-Ripoll et al. [40] propose visualization diagrams for TT-format surrogates of several mechanical simulations, emphasizing multidimensionality and real-time reconstruction.

A few papers have extracted Sobol indices from TT surrogates. Dolgov et al. [14] build their decomposition using ACA and derive properties and statistics including means, covariances, level sets, and individual Sobol indices. Zhang et al. [41] developed a hierarchical uncertainty quantification algorithm using TT and PCE to estimate a circuit's response depending on its subcomponents' behavior. Rai [13] gives formulas to compute Sobol indices from a range of low-rank approximation surrogates, including TT-based.

## 4. Construction of TT Surrogates

Surrogate-based sensitivity analysis methods are only as good as the approximant's accuracy w.r.t. to the true unknown model. A key part of our pipeline is thus obtaining a high-quality TT interpolant. Fortunately, many models can be accurately represented by a low-rank TT model. For example, multiplicative functions (i.e. with the form $f_1(x_1) \cdots f_N(x_N)$) have exactly rank 1, while additive terms (i.e. $f_1(x_1) + ... + f_N(x_N)$) have exactly rank 2. More generally, we can build TT surrogates in a wide range of settings.

### 4.1. Preliminaries: Variable Range Discretization

The methods we present are applicable to both continuous and categorical variables, and these two kinds may coexist within one model. However, in order to build the tensor product grid $I_1 \times ... \times I_N$ for our variable space, our TT surrogate $\tilde{f}(\boldsymbol{x}) \approx f(\boldsymbol{x})$ needs to discretize each continuous variable's domain as a finite set $x_n(1) < ... < x_n(I_n)$. To record or evaluate an entry $\boldsymbol{x}$, each coordinate $x_n$ must be first quantized to match the corresponding axis discretization. This is not a problem in practice, and discretizing the variable space is indeed a usual feature of several sampling strategies such as factorial design, Morris' method, one-at-a-time design, etc. [5]. If needed, the grid can be refined by simply increasing the sampling resolution before building the surrogate, and all important TT operations are linear w.r.t. the spatial dimensions $I_n$. For simplicity we use nearest-neighbor interpolation to convert an arbitrary $\boldsymbol{x}$ to integer tensor indices $1, ..., I_n$.

### *4.2. Construction From a Black-Box System*

*Black-box adaptive sampling* is the scenario in which new samples $\mathbf{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_P\}$ are to be chosen and evaluated from scratch with no prior information on the inner workings of the true model. One has freedom to select the set of samples $\mathbf{X}$, and can do so adaptively in order to minimize the model's generalization error. Adaptive cross approximation (ACA) builds a progressive sampling plan on the low-rank assumption; it is an example of design of experiments (DOE). ACA has long been an active research topic [42, 43, 30, 31, 21] and has become a key tool to create and manipulate tensors. Recent techniques generalize the maximum-volume (*maxvol* for short) algorithm, which approximates a matrix in terms of a carefully selected subset of its rows and columns. In higher dimensions, ACA constructs the plan by progressively sampling certain *tensor fibers*: sets of samples obtained by fixing all parameters but one. See Fig. 3 for an example.

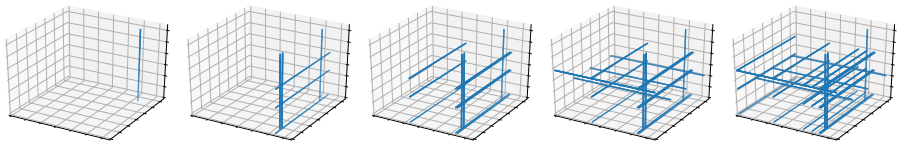


Figure 3: Example progression of the sampling plan during an ACA for a 3D tensor.

This is a case of one-at-a-time (OAT) sampling, a strategy known to potentially improve the DOE's overall efficiency (see also [1], 2.4.2). Like Latin hypercube sampling, this guarantees that all possible discretized values for every variable are used at least once. In this paper we use an *alternating minimal energy* method to select the fibers [44], an algorithm whose implementation has publicly been released as part of the Python ttpy toolbox [36].

### *4.3. From Categorical Data*

Tensors are discrete data structures indexed by discrete axes and thus support categorical variables in a natural way (consider for example the 2D case: the rank of a matrix is not affected if we permute its columns or rows). Populating the missing entries of a tensor without any prior assumption about smoothness is known as *tensor completion* and is a very convenient tool for regression on categorical variables. It is similar to the better known problem of low-rank matrix completion for $N = 2$, but specific algorithms for $N \geq 3$ of course depend heavily on the particular decomposition format chosen (CP, Tucker, TT, etc.). We have implemented an *alternating least squares* (ALS) completion algorithm in the TT format [45] and used it to learn a 14D categorical data set (see Sec. 8.3). The algorithm is a form of block coordinate descent, whereby one TT core is optimized at a time and the relative error is provably non-increasing.

*4.4. From an Auxiliary Regressor*

More generally, one may want to interpolate the given training set first with a preferred regression method: support vector machines, radial basis functions, Gaussian processes, etc. One can then approximately transform this auxiliary surrogate into the TT representation by an ACA algorithm. This is a very general approach and is feasible as long as the intermediate regression's output can be approximated well by a surrogate of low TT ranks. Under this assumption, ACA works as a universal tool to reduce any model into the TT format, usable whenever an ad-hoc conversion in the compressed domain (such as the ones discussed next) is not available.

*4.5. From Another Low-Rank Decomposition*

Several well-known surrogate models are actually based on a low-rank expression, or can easily be cast as a low-rank format. We can convert from these cases more directly instead of relying on the general ACA as just discussed in Sec. 4.4.

*4.5.1. From CP*

TT ranks are bounded from above by CP ranks [12], and the proof is constructive: Given a rank-$R$ CP decomposition $[[\mathbf{U}^{(1)}, ..., \mathbf{U}^{(N)}]]$, an equivalent TT expression $[[\mathcal{T}^{(1)}, ..., \mathcal{T}^{(N)}]]$ can be straightforwardly built as:

$$\mathcal{T}^{(n)}[x_k] := \mathrm{diag}(\mathbf{U}^{(n)}[x_k, :]), k = 1, ..., I_n \tag{13}$$

where the $n$-th core has size $R \times I_n \times R$. Thanks to this we can convert an arbitrary low-rank CP surrogate into our preferred standard TT representation.

*4.5.2. From Tucker*

A TT can be also obtained from a Tucker decomposition, although TT-ranks are not bounded by Tucker ranks (and vice versa). To do this conversion we start with the Tucker approximation formula

$$\tilde{f} \approx \mathcal{T} = [[\mathcal{B}; \mathbf{U}^{(1)}, ..., \mathbf{U}^{(N)}]] \tag{14}$$

and then compress its core in the TT format:

$$\mathcal{T} \approx [[\, [[\mathcal{B}^{(1)}, ..., \mathcal{B}^{(N)}]]; \mathbf{U}^{(1)}, ..., \mathbf{U}^{(N)}]] \tag{15}$$

By multilinearity, the right-hand side of Eq. 15 equals the following expression

$$[[\mathcal{B}^{(1)} \times_2 \mathbf{U}^{(1)}, ..., \mathcal{B}^{(N)} \times_2 \mathbf{U}^{(N)}]] \tag{16}$$

Eq. 16 is a so-called *TT-Tucker* decomposition and was originally considered in [46]. See also Fig. 4 for a graphical representation in terms of tensor networks, similarly to [32].

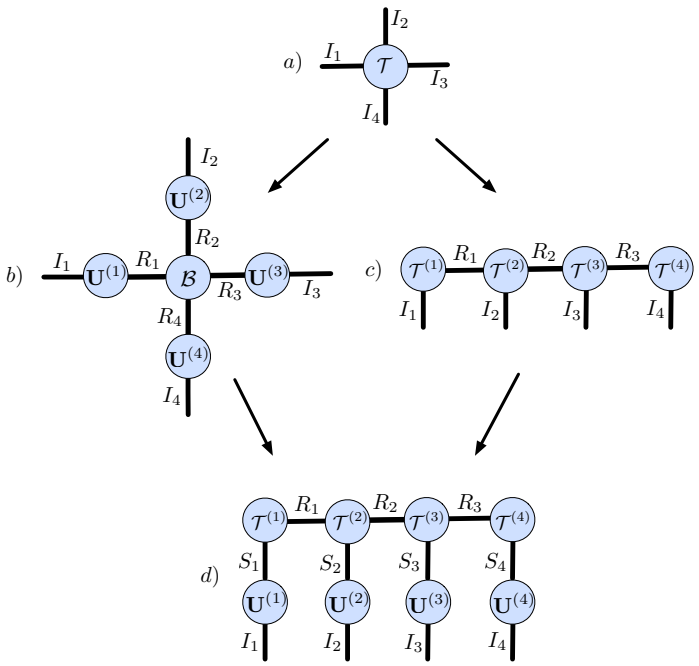



Figure 4: a) A full 4D tensor $\mathcal{T}$; b) in Tucker format; c) in TT format; d) in TT-Tucker format. The TT-Tucker representation can be computed in two alternative but equivalent ways: either via TT compression of the Tucker core $\mathcal{B}$ (left), or via Tucker compression (along the 2nd mode) of each individual TT core $\mathcal{T}^{(n)}$ (right). Similarly, the Tucker format may be cast to TT by following either the path b)-a)-c) (full decompression and compression) or the much less expensive b)-d)-c).

The final TT cores are retrieved by explicitly performing the tensor-times-matrix operations:

$$\mathcal{T}^{(n)} = \mathcal{B}^{(n)} \times_2 \mathbf{U}^{(n)} \tag{17}$$

which increases the overall size, but is still linear w.r.t. $N$.

*4.5.3. From Polynomial Chaos Expansions*

PCE surrogate models have been used in stochastic modeling and uncertainty quantification for decades. A PCE is based on a set of $N$ polynomial bases $\mathcal{P}^{(1)}, ..., \mathcal{P}^{(N)}$ with each basis $\mathcal{P}^{(n)} = \{\mathcal{P}_0^{(n)}, \mathcal{P}_1^{(n)}, ...\}$ being orthogonal w.r.t. $x_n$'s marginal PDF $dF_n$:

$$\int_{\Omega_n} \mathcal{P}_i^{(n)}(x_n)\mathcal{P}_j^{(n)}(x_n)\, dF_n(x_n) = 0 \ \forall n \text{ iff } i \neq j \tag{18}$$

The PCE of bounded degree $D$ approximates a function $f : \Omega \subset \mathbb{R}^N \to \mathbb{R}$

as a truncated expansion in terms of these bases:

$$f(\boldsymbol{x}) \approx \sum_{\boldsymbol{\alpha}=(0,\dots,0)}^{(D,\dots,D)} \mathcal{C}_{\boldsymbol{\alpha}} \cdot \Psi_{\boldsymbol{\alpha}}(\boldsymbol{x}) \tag{19}$$

with

$$\Psi_{\boldsymbol{\alpha}}(\boldsymbol{x}) := \prod_{n=1}^{N} \mathcal{P}_{\alpha_n}^{(n)}(x_n) \tag{20}$$

being $\mathcal{C}$ an $N$-dimensional tensor containing the expansion weights.

Sudret [2] established a connection between the Sobol decomposition and the PCE that has gained significant popularity [47, 17, 14]. The author proposed the indices $SU_{\boldsymbol{\alpha}}$, which approximate each Sobol coefficient $S_{\boldsymbol{\alpha}}$ from a PCE surrogate of bounded degree. The idea behind $SU$, similar to [48], is based on the fact that the first function $\mathcal{P}_0^{(n)}$ from every PCE basis is a zero-degree polynomial and therefore a constant. From this and the basis' orthogonality it follows that the fraction of the model's response that is *not* explained by a specific variable $n$ is exactly the one captured by the projection onto $\mathcal{P}_0^{(n)}$, while the remaining $\{\mathcal{P}_d^{(n)}\}_{d\geq 1}$ account for the interactions where the variable is present.

We can convert any such PCE representation into a TT surrogate as follows, at the expense of only a small discretization error (that can be easily adjusted, recall Sec. 4.1). Eq. 19 is interpretable as a *continuous* Tucker decomposition, with the $f_{\boldsymbol{\alpha}}$ acting as the elements of a core of size $(D+1)^N$. To obtain a standard Tucker format we just need to define its factor matrices. Each factor $\mathbf{U}^{(n)}$ has size $I_n \times (D+1)$ and is found by sampling the corresponding polynomial basis over the discretized variable range $x_n(1), ..., x_n(I_n)$:

$$\mathbf{U}^{(n)}[i,j] := \mathcal{P}_j^{(n)}(x_n(i)) \tag{21}$$

After this we can apply the conversion method detailed in Sec. 4.5.2 to get the equivalent TT representation.

Alternatively, we may also convert a low-rank PCE expansion [11] to TT by means of the CP conversion method above. Such low-rank expansions define a *continuous* CP as

$$f(\boldsymbol{x}) \approx \sum_{r=1}^{R} \lambda_r \cdot \Psi_r(\boldsymbol{x}) \tag{22}$$

with each rank-1 component being the outer product of functions that admit a low-degree polynomial expansion:

$$\Psi_r(\boldsymbol{x}) := \prod_{n=1}^{N} \left( \sum_{d=0}^{D} \alpha_{rd}^{(n)} \cdot \mathcal{P}_d^{(n)}(x_n) \right) \tag{23}$$

and can be converted into standard PCE via discretization, analogously to Eq. 21.

## 5. The Sobol Tensor Train

We now introduce our proposed *Sobol tensor train*, denoted as $\mathcal{S}$, which has dimension $N$ and size 2 along each dimension for a total of $2^N$ elements. Such $2 \times ... \times 2$ tensors are not unusual, see for example the so-called *quantized tensor train* (QTT) and the closely-related *wavelet tensor train* (WTT) [49], as well as the recent multilinear regressors known as *exponential machines* [50]. $\mathcal{S}$ hence records the variance components for all $n$-ary interactions:

$$S_{\boldsymbol{\alpha}} \approx \mathcal{S}_{\boldsymbol{\alpha}} = \mathcal{S}[j_1, \dots, j_N] = \mathcal{S}^{(1)}[j_1] \cdot ... \cdot \mathcal{S}^{(N)}[j_N] \tag{24}$$

with $j_n = 1$ if $n \in \boldsymbol{\alpha}$ and 0 otherwise. To construct it we combine the definitions of Sobol decomposition (Sec. 2.1) and variance components (Sec. 2.2) with the TT formulation as follows.

**Proposition 5.1.** *Let $\mathbf{x} = (x_1, ..., x_N)$ be independent with distributions $F_1, ..., F_N$, and let $\mathcal{T} = [[\mathcal{T}^{(1)}, ..., \mathcal{T}^{(N)}]]$ be a TT surrogate $\tilde{f}(\mathbf{x}) \approx f(\mathbf{x})$. Then, each term $\tilde{f}_{\boldsymbol{\alpha}}$ of the Sobol decomposition of $\tilde{f}$ is given by $\mathcal{T}_{\boldsymbol{\alpha}} = [[\mathcal{T}_{\boldsymbol{\alpha}}^{(1)}, ..., \mathcal{T}_{\boldsymbol{\alpha}}^{(N)}]]$ with cores defined slice-wise as*

$$\mathcal{T}_{\boldsymbol{\alpha}}^{(n)}[j] := \begin{cases} \mathrm{E}[\mathcal{T}^{(n)}] \text{ if } n \notin \boldsymbol{\alpha} \\ \mathcal{T}^{(n)}[j] - \mathrm{E}[\mathcal{T}^{(n)}] \text{ if } n \in \boldsymbol{\alpha} \end{cases} \tag{25}$$

*for all slices $j = 0, ..., I_n - 1$, where $\mathrm{E}[\mathcal{T}^{(n)}] := \frac{1}{I_n} \cdot \sum_{i=0}^{I_n - 1} F_n(x_n(i))\, \mathcal{T}^{(n)}[i]$ is the expectation along the $n$-th dimension, i.e. the average of the $n$-th core's slices, weighted by the $n$-th PDF term.*

See the Appendix B for a proof. Eq. 25 can be intuitively interpreted as follows: Variables not in $\boldsymbol{\alpha}$ must be integrated over their domain of existence, and $\tilde{f}_{\boldsymbol{\alpha}}$ does not effectively depend on them. Their corresponding cores are accordingly constant. For variables in $\boldsymbol{\alpha}$, on the other hand, we must keep the original function but subtract from it the lower-order expectations; these are all correctly accounted for thanks to multilinearity.

With Eq. 25 we can already compute any arbitrary variance component $S_{\boldsymbol{\alpha}}$. However, we give now a more expeditious method that allows us to produce *all* indices at once. First note that the following tensor $\mathcal{T}_* = [[\mathcal{T}_*^{(1)}, \dots, \mathcal{T}_*^{(N)}]]$ compactly encodes all $2^N$ Sobol decomposition terms:

$$\begin{cases} \mathcal{T}_*^{(n)}[0] := \mathrm{E}[\mathcal{T}^{(n)}] \\ \mathcal{T}_*^{(n)}[j] := \mathcal{T}^{(n)}[j-1] - \mathrm{E}[\mathcal{T}^{(n)}] \text{ for } j = 1, ..., I_n \end{cases} \tag{26}$$

This tensor $\mathcal{T}_*$ is simply a concatenation of the two types of slices of Eq. 25, and so it approximates $f_{\boldsymbol{\alpha}}(\boldsymbol{x})$ for any $\boldsymbol{\alpha}$ and any $\boldsymbol{x} \in \Omega$.

We have now all necessary components to construct our Sobol tensor $\mathcal{S}$: we need to compute $\mathrm{V}[\tilde{f}_{\boldsymbol{\alpha}}]/D = \mathrm{E}[(\tilde{f}_{\boldsymbol{\alpha}} - \mathrm{E}[\tilde{f}_{\boldsymbol{\alpha}}])^2]/D$ in the compressed domain. The procedure, detailed in Alg. 1, also obtains the variance indices $D_{\boldsymbol{\alpha}}$ as a necessary subproduct prior to normalization by the total variance $D$.

**Algorithm 1** Given a TT surrogate $\mathcal{T} = [[\mathcal{T}^{(1)}, ..., \mathcal{T}^{(N)}]]$ of size $I_1 \times ... \times I_N$, extract the compressed Sobol tensor $\mathcal{S}$ of size $2 \times ... \times 2$.

1: Compute $\mathcal{T}_*$ as in Eq. 26 $\{\mathcal{T}_*$ encodes $\tilde{f}_{\boldsymbol{\alpha}} \, \forall \boldsymbol{\alpha}\}$
2: Compute $\mathcal{T}_{**} := \mathcal{T}_* \circ \mathcal{T}_* = \mathcal{T}_*^2$ $\{\mathcal{T}_{**}$ encodes $\tilde{f}^2_{\boldsymbol{\alpha}} \, \forall \boldsymbol{\alpha}\}$
3: **for** $n = 1, \ldots, N$ **do**
4: $\quad \mathcal{D}^{(n)}[0] := \mathcal{T}_{**}^{(n)}[0]$
5: $\quad \mathcal{D}^{(n)}[1] := \frac{1}{I_n} \cdot \sum_{i=0}^{I_n - 1} F_n(x_n(i)) \, \mathcal{T}_{**}^{(n)}[i+1]$
6: **end for**
7: $\mathcal{D} := [[\mathcal{D}^{(1)}, ..., \mathcal{D}^{(N)}]]$ $\{\mathcal{D}$ encodes $\mathrm{V}[f_{\boldsymbol{\alpha}}] \, \forall \boldsymbol{\alpha}\}$
8: $D := \prod_{n=1}^{N} (\mathcal{D}^{(n)}[0] + \mathcal{D}^{(n)}[1])$ {Total variance $\mathrm{V}[\tilde{f}]$}
9: $\mathcal{S} := \mathcal{D}/D$
10: **return** $\mathcal{S}$

If the input surrogate has TT-ranks $R_1, ..., R_{N-1}$, then $\mathcal{S}$ may have at most ranks $R_1^2, ..., R_{N-1}^2$. The squaring (line 2 from Alg. 1) can be achieved either by ACA or by slice-wise Kronecker product if the rank is low enough (see Appendix A). All other operations cannot increase any of the ranks. Last, note that the first corner coefficient in the tensor $\mathcal{S}_\emptyset = \mathcal{S}[0, \ldots, 0] = \tilde{f}^2_\emptyset / D$ is not a variance component; we set it to zero if needed with a simple rank-1 correction:

$$\mathcal{S} \leftarrow \mathcal{S} - \begin{pmatrix} \tilde{f}^2_\emptyset / D \\ 0 \end{pmatrix} \otimes \overbrace{\begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes ... \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix}}^{N-1 \text{ terms}} \tag{27}$$

## 6. Aggregating Indices

Aggregated indices require up to an exponential number of addends if computed naively. But thanks to multilinearity of the proposed tensor decomposition, we can obtain all such QOI at once and at very little cost.

### *6.1. Superset Sobol Tensors*

We recall now the notion of superset indices from Sec. 2.3, which capture the aggregate dependence with respect to a group of indices; i.e. sum the variance component of $\boldsymbol{\alpha}$ with those of all tuples that are a superset of $\boldsymbol{\alpha}$. If $\mathcal{S}$ is available, we can construct a superset Sobol tensor $\mathcal{S}^S = [[\mathcal{S}^{S(1)}, ..., \mathcal{S}^{S(N)}]]$ that approximates any $S^S_{\boldsymbol{\alpha}} \approx \mathcal{S}^S_{\boldsymbol{\alpha}}$. We construct its cores by slice-wise manipulation of the original cores:

$$\begin{cases} \mathcal{S}^{S(n)}[0] := \mathcal{S}^{(n)}[0] + \mathcal{S}^{(n)}[1] \\ \mathcal{S}^{S(n)}[1] := \mathcal{S}^{(n)}[1] \end{cases} \tag{28}$$

The rationale is that variables that are present in a tuple (encoded by the second slices, $j = 1$) should stay present, while the rest (first slices, $j = 0$) should be accounted for both when they are absent and when they are included.

As an example, let us consider in 2D the second superset index $\mathcal{S}^S_2 := \mathcal{S}_{12} + \mathcal{S}_2$. Eq. 28 yields

$$\begin{aligned} \mathcal{S}^S_2 &= \mathcal{S}^{S(1)}[0] \cdot \mathcal{S}^{S(2)}[1] = (\mathcal{S}^{(1)}[0] + \mathcal{S}^{(1)}[1]) \cdot \mathcal{S}^{(2)}[1] \\ &= \mathcal{S}^{(1)}[0] \cdot \mathcal{S}^{(2)}[1] + \mathcal{S}^{(1)}[1] \cdot \mathcal{S}^{(2)}[1] = \mathcal{S}_2 + \mathcal{S}_{12} \end{aligned} \tag{29}$$

as expected. Conversely, one may extract $\mathcal{S}$ from $\mathcal{S}^S$ by reverting the slice-wise transformations:

$$\begin{cases} \mathcal{S}^{(n)}[0] = \mathcal{S}^{S(n)}[0] - \mathcal{S}^{S(n)}[1] \\ \mathcal{S}^{(n)}[1] = \mathcal{S}^{S(n)}[1] \end{cases} \tag{30}$$

We wish to emphasize the compactness and convenience of the relations given by Eqs. 28 and 30. A naive sum to obtain a superset index of order $K$ out of the standard indices $\mathcal{S}$ would require $2^{N-K}$ addends. For example, for $N = 3$ and $\boldsymbol{\alpha} = \{1\}$ we have $\mathcal{S}^S_1 = \mathcal{S}_1 + \mathcal{S}_{12} + \mathcal{S}_{13} + \mathcal{S}_{123}$. Conversely, producing indices $\mathcal{S}$ from $\mathcal{S}^S$ needs $2^{N-K}$ mixed additions and subtractions as dictated by the inclusion-exclusion principle from combinatorics. For instance, $\mathcal{S}_1 = \mathcal{S}^S_1 - \mathcal{S}^S_{12} - \mathcal{S}^S_{13} + \mathcal{S}^S_{123}$. On the other hand, Eqs. 28 and 30 need only $O(NR^2)$ operations in the TT format.

### *6.2. Closed Sobol Tensors*

Similarly to Eq. 28, we derive the closed Sobol tensor $\mathcal{S}^C$ from $\mathcal{S}$ as follows:

$$\begin{cases} \mathcal{S}^{C(n)}[0] := \mathcal{S}^{(n)}[0] \\ \mathcal{S}^{C(n)}[1] := \mathcal{S}^{(n)}[0] + \mathcal{S}^{(n)}[1] \end{cases} \tag{31}$$

The logic here is that indices absent in a tuple should stay absent, while present indices should be accounted for also when they are missing (since we want to sum all subsets). The converse equation reads

$$\begin{cases} \mathcal{S}^{(n)}[0] = \mathcal{S}^{C(n)}[0] \\ \mathcal{S}^{(n)}[1] = \mathcal{S}^{C(n)}[1] - \mathcal{S}^{C(n)}[0] \end{cases} \tag{32}$$

### *6.3. Total Sobol Tensors*

Our last aggregated tensor is the total $\mathcal{S}^T$ and can be obtained via the complement operation as $\mathcal{S}^T_{\boldsymbol{\alpha}} = 1 - \mathcal{S}^C_{-\boldsymbol{\alpha}}$. Let us define a complement tensor $\bar{\mathcal{S}}^C$, defined for each tuple as $\bar{\mathcal{S}}^C_{\boldsymbol{\alpha}} := \mathcal{S}^C_{-\boldsymbol{\alpha}}$. We extract this tensor from $\mathcal{S}^C$ by simply swapping the two slices of each core:

$$\begin{cases} \bar{\mathcal{S}}^{C(n)}[0] := \mathcal{S}^{C(n)}[1] \\ \bar{\mathcal{S}}^{C(n)}[1] := \mathcal{S}^{C(n)}[0] \end{cases} \tag{33}$$

and the final result $\mathcal{S}^T = 1 - \bar{\mathcal{S}}^C$ follows from a simple tensor-tensor subtraction. To retrieve $\mathcal{S}^C$ back from $\mathcal{S}^T$ it suffices to repeat the whole transformation.

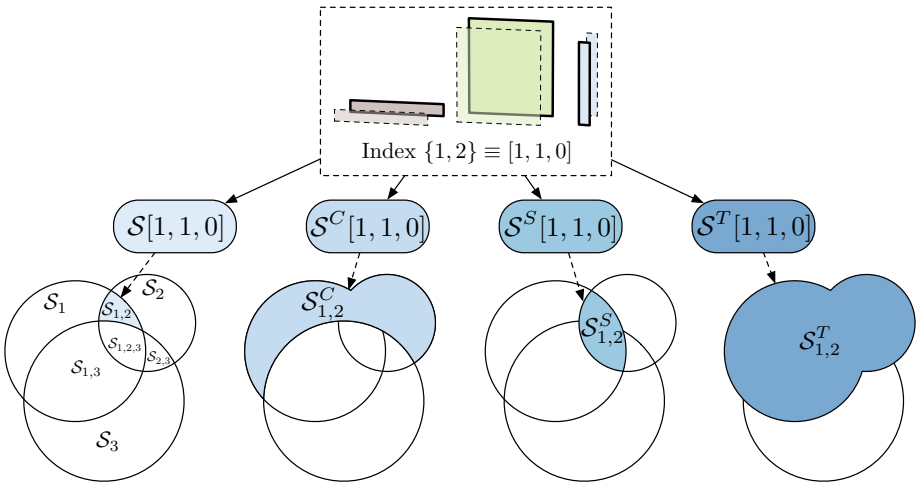



Figure 5: Examples of variance components $\mathcal{S}$, closed $\mathcal{S}^C$, superset $\mathcal{S}^S$, and total Sobol indices $\mathcal{S}^T$ for a 3-variable model, interpreted as set cardinalities. Each colored region area is obtained from its corresponding tensor by multiplying together the indexed slices.

## 7. Global Sensitivity Metrics and Queries

### *7.1. Relevant Subsets of Variables*

A typical and fundamental target in SA is to "*select the $k$ variables that account for the most variance*", or alternatively "*select the smallest set of variables that account for at least (say) 99% variance*". In order to tackle this we introduce the *Hamming mask* of order $k$, that we define as

$$\mathcal{M}^k_{\boldsymbol{\alpha}} := \begin{cases} 1 & \text{if } |\boldsymbol{\alpha}| = k \\ 0 & \text{otherwise} \end{cases} \tag{34}$$

We are able to build its compressed version using only $k+1$ ranks (Fig. 6). It is best understood by following the vector-matrix sequence of products that takes place to reconstruct one element, left to right. The vector at the $n$-th step has size $k+1$ and encodes how many '1' bits have been encountered so far: a '1' at its first position means 0, a '0' followed by a '1' means 1, etc. The core slices transform this vector counter to account for the new bits as we traverse the binary sequence. The first slice of each core is the identity matrix, since it corresponds to a bit set to 0 (which does not have an effect). The second slice, however, must increment the counter, i.e. shift the '1' one position towards the right. It is therefore implemented as a shifted identity matrix. The last core simply checks if the total number of '1' found until the end matches $k$ or not.

The mask tensor $\mathcal{M}_{\boldsymbol{\alpha}}$ allows us to define restricted searches. For instance,

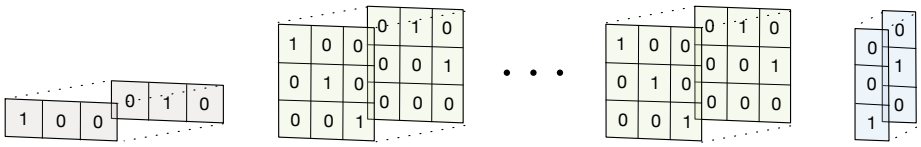


Figure 6: The Hamming mask tensor train $\mathcal{M}^k$ for order $k = 2$. At each position $\boldsymbol{\alpha} \in \{0,1\}^N$ it contains a '1' if and only if $|\boldsymbol{\alpha}| = k$, and 0 otherwise. It is compressed with $N$ cores (rank $k+1$) using $2(k+1)^2(N-2) + 4(k+1)$ elements in total.

the single most important variance component of order $k$ is

$$\arg\max_{\boldsymbol{\alpha}} \left\{ (\mathcal{S} \circ \mathcal{M}^k)_{\boldsymbol{\alpha}} \right\} \tag{35}$$

which we solve using a state-of-the-art global optimization algorithm in the TT format (Sec. 3.3.2). One may also use $\mathcal{S}^T, \mathcal{S}^C$ or $\mathcal{S}^S$ instead of $\mathcal{S}$ depending on the task at hand. For example, the $\mathcal{S}_{\boldsymbol{\alpha}}$ do play the dominant role in factor prioritization, but for factor fixing one is advised to seek a tuple with the smallest total index [1].

We also use $\mathcal{M}^k$ to compute the overall per-order contributions: the tensor dot product

$$< \mathcal{S}, \mathcal{M}^k > \tag{36}$$

gives us the combined order $k$ indices $\sum_{|\boldsymbol{\alpha}|=k} \mathcal{S}_{\boldsymbol{\alpha}}$.

### *7.2. Other Constraints*

The analyst may seek a model simplification that satisfies additional constraints, e.g. that certain variables must, or must not, become frozen. Such conditions can be easily imposed by editing the mask tensor $\mathcal{M}$. For instance, if a variable $1 \leq n \leq N$ should be fixed (i.e. simplified) it is sufficient to remove the second slice of the $n$-th mask core. This effectively restricts the search to $\{\boldsymbol{\alpha} \mid n \notin \boldsymbol{\alpha}\}$ as desired. Conversely, if we wish to ensure that a variable is not fixed (i.e. remains active in the new simplified model), we just remove the first slice of the $n$-th core of $\mathcal{M}$.

## 8. Experimental Results

Our experiments were conducted in Python. We exploit the ttpy toolbox[36], a Python/Fortran library for TT manipulation that supports, among others, compression from full explicit tensors, slicing, decompression, truncation (rounding), and cross-approximation for any dimensionality.

### 8.1. Sobol "G" Function

This function has been extensively used in the SA literature owing to its flexibility and relatively high-order interactions. It is defined as

$$f(\boldsymbol{x}) := \prod_{n=1}^{N} \frac{|4x_n - 2| + a_n}{1 + a_n} \tag{37}$$

being $a_i$ random coefficients sampled from a uniform distribution $\mathcal{U}(0,1)$ and $x_n \sim \mathcal{U}(0,1)$ the $n$ function parameters. Note that $f$ is non-differentiable at one point, namely $(0.5, ..., 0.5)$. We can expect to get an exact TT interpolator (up to round-off error) of $f$ as it is a product of univariate functions and therefore it has rank 1. Our test example uses $N = 25$ dimensions. We discretize each variable into $I_1 = \cdots = I_{25} = 64$ possible values, namely $\{0, \frac{1}{63}, \ldots, \frac{62}{63}, 1\}$, in order to obtain a tensor grid containing $64^{25} \approx 1.4 \cdot 10^{45}$ elements. To build our compressed TT surrogate we ran an ACA-driven sampling plan using 3200 evaluations of $f$ placed on this grid. Fig. 7 illustrates how the 3200 samples were placed along the 25 dimensions; the first and last bins were visited the most.

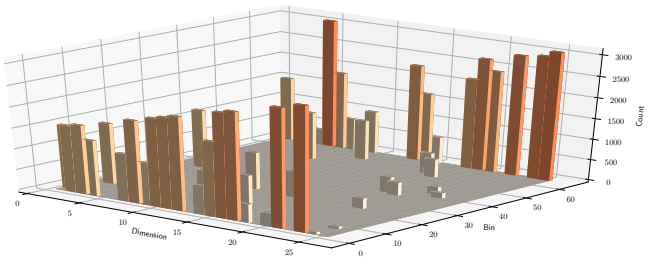



Figure 7: Sample placement frequencies recorded while fitting a TT surrogate of the Sobol "G" function using ACA, broken down by dimension and bin.

We tested the resulting model over a test set of 4096 samples drawn at random using a Latin hypercube sampling (LHS) scheme, where it achieved a relative error of $\epsilon \approx 4.646 \cdot 10^{-15}$. Extracting the Sobol TT from the surrogate took 4.93 seconds using $\epsilon = 10^{-6}$ as the ACA relative error for the squaring step in Alg. 1. Fig. 8 shows every value of $a_n$ and its corresponding first-order Sobol index, computed using our method (Alg. 1).

Tab. 1 shows the 5 highest variance components of any order from our method, which in this case are only order-1 effects. We also show the indices as estimated directly from sampling the TT surrogate via the Sensitivity Analysis Library (SALib [51]) in Python, using quasi-MC with varying number of sample points $P$. Finally and to complete the cross-check, we list the analytical Sobol values for comparison [52]: $D_n = 1/(3(1 + a_n)^2), D = \prod_n (D_n + 1) - 1$, and $S_n = (\prod_n D_n)/D$. Tab. 2 shows the highest aggregated indices (i.e. total, closed and superset) of order 1, 2, and 3 separately.

We observe that the TT indices are accurate to almost 4 decimal digits. The indices computed by SALib become closer the more samples are taken, further

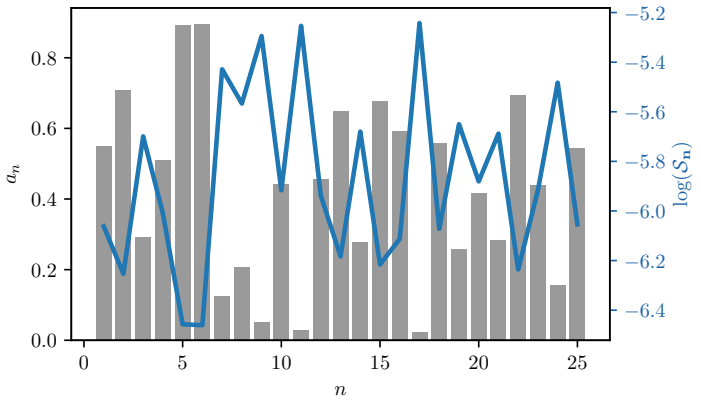



Figure 8: The 25 random values chosen for $a_n$ and their resulting Sobol values (depicted in logarithmic scale).

Table 1: Highest variance components for the Sobol G function

| **Index** | **Value** | | | | |
|---|---|---|---|---|---|
| | Sobol TT | SALib | | | Analytical |
| | | $(P = 520000)$ | $(P = 5.2 \cdot 10^6)$ | $(P = 5.2 \cdot 10^7)$ | |
| $\mathcal{S}_{17}$ | 0.0053 | 0.0147 | 0.0110 | 0.0073 | 0.0054 |
| $\mathcal{S}_{11}$ | 0.0052 | -0.0001 | 0.0053 | 0.0052 | 0.0054 |
| $\mathcal{S}_9$ | 0.0050 | 0.0237 | 0.0084 | 0.0053 | 0.0051 |
| $\mathcal{S}_7$ | 0.0044 | -0.0038 | 0.0035 | 0.0039 | 0.0045 |
| $\mathcal{S}_{24}$ | 0.0042 | 0.0259 | 0.0062 | 0.0042 | 0.0042 |

supporting the correctness of our method. Note that for this function SALib required a very large number of samples to reach a similar level of precision, in contrast to our proposed method. We attribute this to the function's high dimensionality, which can be handled well by the TT.

### *8.2. Piston Simulation*

This is a lower-dimensional but more complex model that measures the cycle time of a piston simulation [53]. The output is defined analytically on 7 variables as

$$f(\boldsymbol{x}) = 2\pi \sqrt{\frac{M}{k + S^2 \frac{P_0 V_0}{T_0} \frac{T_a}{V^2}}} \tag{38}$$

Table 2: Highest aggregated indices of order 1, 2, and 3 for the Sobol G function

| **Order** | **Index / Variable(s)** | | |
|---|---|---|---|
| | Total | Closed | Superset |
| 1 | $\mathcal{S}^T_{17} = 0.2452$ | $\mathcal{S}^C_{17} = 0.0053$ | $\mathcal{S}^S_{17} = 0.2452$ |
| 2 | $\mathcal{S}^T_{11,17} = 0.4296$ | $\mathcal{S}^C_{11,17} = 0.0122$ | $\mathcal{S}^S_{11,17} = 0.0586$ |
| 3 | $\mathcal{S}^T_{9,11,17} = 0.5657$ | $\mathcal{S}^C_{9,11,17} = 0.0209$ | $\mathcal{S}^S_{9,11,17} = 0.0136$ |

with

$$V = \frac{S}{2k}\left(\sqrt{A^2 + 4k\frac{P_0 V_0}{T_0}T_a} - A\right)$$
$$A = P_0 S + 19.62M - \frac{kV_0}{S} \tag{39}$$

Table 3: Parameters of the Piston simulation

| **Variable** | **Description** | **Units** | **Distribution** |
|---|---|---|---|
| $M$ | Piston weight | $kg$ | $\mathcal{U}(30, 60)$ |
| $S$ | Piston surface area | $m^2$ | $\mathcal{U}(0.005, 0.02)$ |
| $V_0$ | Initial gas volume | $m^3$ | $\mathcal{U}(0.002, 0.01)$ |
| $k$ | Spring coefficient | $N/m$ | $\mathcal{U}(1000, 5000)$ |
| $P_0$ | Atmospheric pressure | $N/m^2$ | $\mathcal{U}(90000, 110000)$ |
| $T_a$ | Ambient temperature | $K$ | $\mathcal{U}(290, 296)$ |
| $T_0$ | Filling gas temperature | $K$ | $\mathcal{U}(340, 360)$ |

The full list of parameters and their input ranges is detailed in Tab. 3. Our model was generated with ACA, stopped after 43904 function evaluations, again with $I = 64$ bins per dimension. It has 10496 non-zero elements and maximum rank $R = 7$, and it achieves $\epsilon \approx 0.077\%$ over an LHS-acquired test set. Note that, again, the TT model is built with fewer samples than those needed by SALib's MC algorithm, and that it is able to compute indices of arbitrary order *a posteriori*. Extracting the Sobol TT took 5.70 seconds in this case.

For further comparison we have also computed a PCE approximation of this function via 4096 training samples chosen similarly to the test set. To build the model we take the 4 first Legendre polynomials for each variable. We compress then the PCE-Tucker core into a TT model as detailed in Sec. 4.5 with a relative error of 0.5%, resulting in $R = 22$. The resulting TT-PCE model approximates the training set with a relative error $\epsilon \approx 0.38\%$, and achieves $\epsilon \approx 1.22\%$ on the test set.

As shown in Tab. 4, our analysis reveals the fact that only the 4 first variables have a significant first-order effect. Their numerical values are consistent with the results reported e.g. in [4] (after normalization). Also, the most important tuple interactions arise from these very same variables. The triplet $\{S, V_0, k\}$ in particular has a closed index of about 95% as reported in Tab. 5. Overall,

interactions of order 3 and above play a relatively small role. The proposed method is again able to compute all Sobol and aggregated indices in one go, and to do so with fewer evaluations than SALib.

Table 4: Highest variance components for the piston function (interactions of order 3 and above are not supported by SALib)

| **Index** | **Var(s)** | **Value** | | | |
|---|---|---|---|---|---|
| | | Sobol TT | Sobol TT-PCE | SALib on TT ($P$=160000) | SALib ($P$=160000) |
| $\mathcal{S}_2$ | $S$ | 0.5545 | 0.5585 | 0.5562 | 0.5563 |
| $\mathcal{S}_3$ | $V_0$ | 0.3207 | 0.3238 | 0.3215 | 0.3215 |
| $\mathcal{S}_1$ | $M$ | 0.0390 | 0.0396 | 0.0389 | 0.0391 |
| $\mathcal{S}_{2,4}$ | $S$,$k$ | 0.0242 | 0.0211 | 0.0252 | 0.0250 |
| $\mathcal{S}_4$ | $k$ | 0.0212 | 0.0200 | 0.0219 | 0.0221 |
| $\mathcal{S}_{3,4}$ | $V_0$,$k$ | 0.0129 | 0.0117 | 0.0121 | 0.0118 |
| $\mathcal{S}_{2,3,4}$ | $S$,$V_0$,$k$ | 0.0094 | 0.0066 | - | - |
| $\mathcal{S}_{1,3}$ | $M$,$V_0$ | 0.0050 | 0.0046 | 0.0053 | 0.0053 |
| $\mathcal{S}_{2,3}$ | $S$,$V_0$ | 0.0046 | 0.0043 | 0.0045 | 0.0044 |
| $\mathcal{S}_{1,2}$ | $M$,$S$ | 0.0046 | 0.0048 | 0.0036 | 0.0035 |

Table 5: Highest aggregated indices of order 1, 2, and 3 for the piston function

| **Order** | **Index / Variable(s)** | | |
|---|---|---|---|
| | Total | Closed | Superset |
| 1 | $\mathcal{S}_2^T = 0.5987$ $\{S\}$ | $\mathcal{S}_2^C = 0.5545$ $\{S\}$ | $\mathcal{S}_2^S = 0.5987$ $\{S\}$ |
| 2 | $\mathcal{S}_{2,3}^T = 0.9374$ $\{S,V_0\}$ | $\mathcal{S}_{2,3}^C = 0.8799$ $\{S,V_0\}$ | $\mathcal{S}_{2,4}^S = 0.0343$ $\{S,k\}$ |
| 3 | $\mathcal{S}_{1,2,3}^T = 0.9776$ $\{M,S,V_0\}$ | $\mathcal{S}_{2,3,4}^C = 0.9475$ $\{S,V_0,k\}$ | $\mathcal{S}_{2,3,4}^S = 0.0098$ $\{S,V_0,k\}$ |

### *8.3. SGEMM Matrix Product in the GPU*

Our last experiment is a parallel computing example: we measured the computation time of 32-bit floating point matrix-matrix products in a graphics processing unit (GPU) according to 14 parameters and optimization techniques (loop unrolling, thread block-size, vector data types, etc.). The input variables are essentially discrete, since they are highly non-linear [54] and can only take a handful of different values at most (usually a few powers of 2). We have chosen to build our TT surrogate using ALS tensor completion as we described in Sec. 4.3. The product analyzed is $\mathbf{A} \cdot \mathbf{B} = \mathbf{C}$ with all three matrices having size $2048 \times 2048$. We use the highly-tuneable SGEMM kernel provided in the package CLTune [54], a generic auto-tuner for OpenCL kernels written in C++. Tab. 6 summarizes the 14 parameters and their input ranges (see the CLTune paper for further details).

Table 6: The 14 parameters of the SGEMM OpenCL Kernel

| Variable(s) | Description | Domain |
|---|---|---|
| $M_{wg}, N_{wg}$ | Per-matrix 2D tiling at workgroup level | $\{16, 32, 64, 128\}$ |
| $K_{wg}$ | Inner dimension of 2D tiling at workgroup level | $\{16, 32\}$ |
| $M_{dimC}, N_{dimC}$ | Local workgroup size | $\{8, 16, 32\}$ |
| $M_{dimA}, N_{dimB}$ | Local memory shape (when enable) | $\{8, 16, 32\}$ |
| $K_{wi}$ | Kernel loop unrolling factor | $\{2, 8\}$ |
| $M_{vec}, N_{vec}$ | Per-matrix vector widths for loading and storing | $\{1, 2, 4, 8\}$ |
| $M_{stride}, N_{stride}$ | Enable stride for accessing off-chip memory within a single thread | $\{yes, no\}$ |
| $L\$_A, L\$_B$ | Per-matrix manual caching of the 2D workgroup tile | $\{yes, no\}$ |

We generated this data set with a workstation running Ubuntu Linux 16.04, equipped with an Intel Core i5-4690 3.5GHz processor and a GeForce GTX680 GPU with 4GB of memory. Among the 1327104 total possible variable combinations, 241600 are feasible due to various kernel constraints; our data set consists of 12080 samples taken uniformly at random from these (without repetition). Each sample was measured 25 times and averaged in order to reduce noise effects. All SGEMM running times are considered in logarithmic scale both for training and analysis as advised in [55]. We split the data as 70%, 15%, and 15% for training, validation and test, respectively. The best TT surrogate was obtained after 25 ALS iterations, which took 28.7 seconds. It has ranks $R_1 = \cdots = R_{13} = 8$ for a total of 2224 non-zero elements. It achieved a relative error of $\epsilon \approx 3.4\%$ on the test set (see Fig. 9). The final indices (Tables 7 and 8) were computed after re-fitting this best model to the full data set. The Sobol TT took 12.32 seconds to derive from the TT surrogate.

Our results indicate a relatively large presence of high-order interactions; this matches the prior knowledge that GPU kernel optimization is a challenging high-dimensional parameter space, and that the parameters' influences tend to be highly inter-dependent [54]. In particular the most important order-1 Sobol index (from $M_{wg}$) is only about 6%, and all order-1 indices combined explain only less than one fourth of the total model variability. We also use this real-world data set to test our querying routines; we report some sample results in Tab. 9 involving various aggregated indices.

To conclude this section we show in Fig. 10 one bar chart per data set, containing the overall relative variance broken down by interaction order.

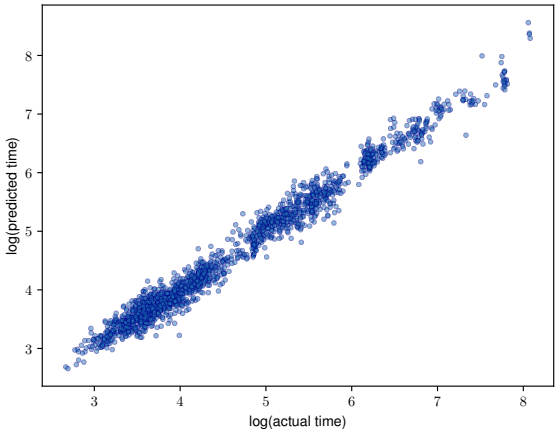
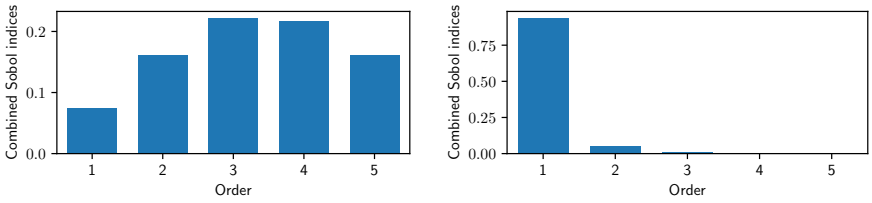
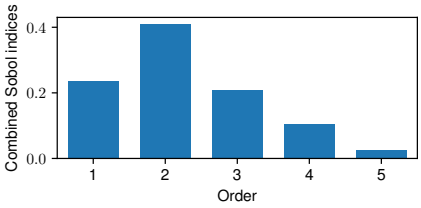



Figure 9: Surrogate obtained via TT completion for our SGEMM experiment: groundtruth vs. prediction over the test set (1812 points), with relative error $\epsilon \approx 3.4\%$




(a) Sobol G function

(b) Piston




(c) SGEMM

Figure 10: Combined contributions for orders 1 to 5 for all three models, efficiently computed using Eq. 36

## 9. Conclusions

We have introduced a compact data structure that gathers all variance components from any TT-based surrogate model and have given algorithms to extract various aggregated Sobol indices from it. The proposed aggregation algorithms capitalize on the format's multilinearity and have very little overhead cost. We combine these ideas with mask tensors, which allow us to

Table 7: Highest variance components for the SGEMM matrix product function

| **Index** | **Var(s)** | **Value** | |
|---|---|---|---|
| | | Sobol TT | SALib on TT ($P$=300000) |
| $\mathcal{S}_{1,2,4}$ | $M_{wg}$,$N_{wg}$,$M_{dimC}$ | 0.0842 | - |
| $\mathcal{S}_{1,4}$ | $M_{wg}$,$M_{dimC}$ | 0.0790 | 0.0799 |
| $\mathcal{S}_{2,4}$ | $N_{wg}$,$M_{dimC}$ | 0.0675 | 0.0754 |
| $\mathcal{S}_1$ | $M_{wg}$ | 0.0643 | 0.0539 |
| $\mathcal{S}_{1,2}$ | $M_{wg}$,$N_{wg}$ | 0.0628 | 0.0686 |
| $\mathcal{S}_{1,4,5}$ | $M_{wg}$,$M_{dimC}$,$N_{dimC}$ | 0.0330 | - |
| $\mathcal{S}_{1,2,4,5}$ | $M_{wg}$,$N_{wg}$,$M_{dimC}$,$N_{dimC}$ | 0.0319 | - |
| $\mathcal{S}_4$ | $M_{dimC}$ | 0.0286 | 0.0257 |
| $\mathcal{S}_{4,5}$ | $M_{dimC}$,$N_{dimC}$ | 0.0225 | 0.0165 |
| $\mathcal{S}_2$ | $N_{wg}$ | 0.0198 | 0.0051 |

Table 8: Highest aggregated indices of order 1, 2, and 3 for the SGEMM matrix product

| **Order** | **Index / Variable(s)** | | |
|---|---|---|---|
| | Total | Closed | Superset |
| 1 | $\mathcal{S}_1^T = 0.6979$ $\{M_{wg}\}$ | $\mathcal{S}_1^C = 0.0643$ $\{M_{wg}\}$ | $\mathcal{S}_1^S = 0.6979$ $\{M_{wg}\}$ |
| 2 | $\mathcal{S}_{1,4}^T = 0.8960$ $\{M_{wg},M_{dimC}\}$ | $\mathcal{S}_{1,4}^C = 0.1718$ $\{M_{wg},M_{dimC}\}$ | $\mathcal{S}_{1,4}^S = 0.4380$ $\{M_{wg},M_{dimC}\}$ |
| 3 | $\mathcal{S}_{1,2,4}^T = 0.9540$ $\{M_{wg},N_{wg},M_{dimC}\}$ | $\mathcal{S}_{1,2,4}^C = 0.4060$ $\{M_{wg},N_{wg},M_{dimC}\}$ | $\mathcal{S}_{1,2,4}^S = 0.2301$ $\{M_{wg},N_{wg},M_{dimC}\}$ |

define restricted queries and thus aid in model reduction/interpretation tasks. We believe the tensor train has a great potential as a canonical format for approximation of multiparametric systems, and the proposed methods for sensitivity analysis can be understood in the context of this trend. The presented framework is flexible in a variety of settings, and supports arbitrary orders of significant variable interactions in higher-dimensional models.

### *9.1. Future Work*

Several possible extensions remain to be explored. For example, Sobol indices based on higher moments [4, 47] are useful for analysis of extreme values and risk minimization, and could be in principle ported to the TT format. Also, we wish to investigate more deeply TT surrogates for multi-valued models. Rather than training a separate model per individual output, we would like to work on a single tensor with an extra dimension to index the outputs. We believe that one may then run a joint TT analysis on the Sobol indices for all outputs at once and thus aid in model interpretability.

Table 9: Once the Sobol TT is available, we can efficiently satisfy various types of queries as detailed in Sec. 7 using constrained search and TT global optimization

| **Query** | **Result** | **Value** | **Computing time (s)** |
|---|---|---|---|
| Variable that interacts the most with $\{L\$_A, L\$_B\}$ | $M_{wg}$ | $\mathcal{S}^S_{1,13,14} = 0.0263$ (as high as possible) | 0.3598 |
| Variable that interacts the least with $M_{wg}$ | $N_{stride}$ | $\mathcal{S}^S_{1,12} = 0.0069$ (as low as possible) | 0.4237 |
| Highest closed 3-tuple that avoids $M_{wg}$ | $N_{wg}$, $M_{dimC}$, $N_{dimC}$ | $\mathcal{S}^C_{2,4,5} = 0.1828$ (as high as possible) | 0.2786 |
| Highest closed 3-tuple that includes $M_{wg}$ | $M_{wg}$, $N_{wg}$, $M_{dimC}$ | $\mathcal{S}^C_{1,2,4} = 0.5940$ (as high as possible) | 0.3686 |
| 6 variables that can be frozen with the least impact | $K_{wg}, K_{wi}, M_{vec}$, $N_{vec}, M_{stride}, N_{stride}$ | $\mathcal{S}^T_{3,8,9,10,11,12} = 0.2365$ (as low as possible) | 0.7307 |

## Appendix A. - Operations in the TT Format

Multiplication/division of a TT tensor by a scalar $\alpha$ is achieved by simply multiplying/dividing one of its cores (say, the first) by $\alpha$. Tensor-tensor addition is written as $(\mathcal{T}_1 + \mathcal{T}_2)[\boldsymbol{x}] := \mathcal{T}_1[\boldsymbol{x}] + \mathcal{T}_2[\boldsymbol{x}]$ and has the following cores:

$$\begin{cases} \left( \begin{array}{c|c} \mathcal{T}_1^{(1)}[x_1] & \mathcal{T}_2^{(1)}[x_1] \end{array} \right) & \text{(first core)} \\ \left( \begin{array}{c|c} \mathcal{T}_1^{(n)}[x_n] & 0 \\ \hline 0 & \mathcal{T}_2^{(n)}[x_n] \end{array} \right) & (1 < n < N) \\ \left( \begin{array}{c} \mathcal{T}_1^{(N)}[x_N] \\ \hline \mathcal{T}_2^{(N)}[x_N] \end{array} \right) & \text{(last core)} \end{cases}$$

The element-wise (or Hadamard) product $(\mathcal{T}_1 \circ \mathcal{T}_2)[\boldsymbol{x}] := \mathcal{T}_1[\boldsymbol{x}] \cdot \mathcal{T}_2[\boldsymbol{x}]$ arises from a slice-wise Kronecker product:

$$(\mathcal{T}_1^{(1)}[x_1] \otimes \mathcal{T}_2^{(1)}[x_1]) \cdot \ldots \cdot (\mathcal{T}_1^{(N)}[x_N] \otimes \mathcal{T}_2^{(N)}[x_N]) \tag{A.1}$$

## Appendix B. - Proof of Proposition 5.1

Consider a tuple $\boldsymbol{\alpha}$ and an arbitrary sampling point $\boldsymbol{x} = (x_1(i_1), \ldots, x_N(i_N))$. We have defined our TT approximation as

$$\mathcal{T}_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \prod_{n=1}^{N} \mathcal{T}_{\boldsymbol{\alpha}}^{(n)}[i_n] \tag{B.1}$$

with

$$\mathcal{T}_{\boldsymbol{\alpha}}^{(n)}[i_n] := \begin{cases} \mathrm{E}[\mathcal{T}^{(n)}] \text{ if } n \notin \boldsymbol{\alpha} \\ \mathcal{T}^{(n)}[i_n] - \mathrm{E}[\mathcal{T}^{(n)}] \text{ if } n \in \boldsymbol{\alpha} \end{cases} \tag{B.2}$$

Expanding the $\boldsymbol{\alpha}$ subtractions from Eq. B.2 we get a sequence of $2^{|\boldsymbol{\alpha}|}$ additions and subtractions:

$$\sum_{\boldsymbol{\beta}\subseteq\boldsymbol{\alpha}} (-1)^{|\boldsymbol{\alpha}|-|\boldsymbol{\beta}|} \prod_{n=1}^{N} \widehat{\mathcal{T}}_{\boldsymbol{\beta}}^{(n)}[i_n] \tag{B.3}$$

with

$$\widehat{\mathcal{T}}_{\boldsymbol{\beta}}^{(n)}[i_n] := \begin{cases} \mathrm{E}[\mathcal{T}^{(n)}] \text{ if } n \notin \boldsymbol{\beta} \\ \mathcal{T}^{(n)}[i_n] \text{ if } n \in \boldsymbol{\beta} \end{cases} \tag{B.4}$$

Recall that $\mathcal{T}^{(n)}$ encodes the model $\tilde{f}$'s response along the $n$-th axis, while $\mathrm{E}[\mathcal{T}^{(n)}]$ represents its integration along that axis. Therefore Eq. B.3 becomes

$$\begin{aligned} &\sum_{\boldsymbol{\beta}\subseteq\boldsymbol{\alpha}} (-1)^{|\boldsymbol{\alpha}|-|\boldsymbol{\beta}|} \int_{\Omega_{-\boldsymbol{\beta}}} \tilde{f}(\boldsymbol{x}) dF_{-\boldsymbol{\alpha}}(\boldsymbol{x}_{-\boldsymbol{\alpha}}) \\ = &\int_{\Omega_{-\boldsymbol{\alpha}}} \tilde{f}(\boldsymbol{x})\, dF_{-\boldsymbol{\alpha}}(\boldsymbol{x}_{-\boldsymbol{\alpha}}) - \sum_{\boldsymbol{\beta}|\boldsymbol{\beta}\subset\boldsymbol{\alpha}} \tilde{f}_{\boldsymbol{\beta}}(\boldsymbol{x}_{\boldsymbol{\beta}}) \\ &= \tilde{f}_{\boldsymbol{\alpha}}(\boldsymbol{x}_{\boldsymbol{\alpha}})\, \square \end{aligned} \tag{B.5}$$


## Acknowledgments

This work was partially supported by the UZH Forschungskredit "Candoc", grant number FK-16-012.


## References

## References


[1] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, S. Tarantola, Global Sensitivity Analysis: The Primer, John Wiley & Sons, Ltd., 2008.

[2] B. Sudret, Global sensitivity analysis using polynomial chaos expansions, Reliability Engineering and System Safety 93 (7) (2008) 964 – 979.

[3] A. Marrel, B. Iooss, B. Laurent, O. Roustant, Calculations of Sobol indices for the gaussian process metamodel, Reliability Engineering and System Safety 94 (2009) 742–751.

[4] A. Owen, J. Dick, S. Chen, Higher Order Sobol' Indices, Information and Inference 3 (2014) 59–81.

[5] B. Iooss, P. Lemaître, A review on global sensitivity analysis methods, in: Uncertainty Management in Simulation-Optimization of Complex Systems: Algorithms and Applications, Springer US, Boston, MA, 2015, pp. 101–122.

[6] I. M. Sobol', Sensitivity estimates for nonlinear mathematical models (in Russian), Mathematical Models 2 (1990) 112–118.

[7] Z. Wu, D. Wang, P. N. Okolo, K. Zhao, W. Zhang, Efficient space-filling and near-orthogonality sequential Latin hypercube for computer experiments, Computer Methods in Applied Mechanics and Engineering 324 (2017) 348–365.

[8] P. Kersaudy, B. Sudret, N. Varsier, O. Piconc, J. Wiart, A new surrogate modeling technique combining Kriging and polynomial chaos expansions – Application to uncertainty analysis in computational dosimetry, Journal of Computational Physics 286 (2015) 103–117.

[9] Z. Wu, D. Wang, P. N. Okolo, F. Hu, W. Zhang, Global sensitivity analysis using a Gaussian radial basis function metamodel, Reliability Engineering and System Safety 154 (2016) 171–179.

[10] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, P. K. Tucker, Surrogate-based analysis and optimization, Progress in Aerospace Sciences 41 (2005) 1–28.

[11] K. Konakli, B. Sudret, Global sensitivity analysis using low-rank tensor approximations, Reliability Engineering & System Safety 156 (2016) 64 – 83.

[12] I. V. Oseledets, Tensor-train decomposition, SIAM Journal on Scientific Computing 33 (5) (2011) 2295–2317.

[13] P. Rai, Sparse Low Rank Approximation of Multivariate Functions – Applications in Uncertainty Quantification, Doctoral thesis, Ecole Centrale Nantes (Nov. 2014).
URL `https://tel.archives-ouvertes.fr/tel-01143694`

[14] S. Dolgov, B. N. Khoromskij, A. Litvinenko, H. G. Matthies, Computation of the response surface in the tensor train data format, arXiv:1406.2816v1 [math.NA] (2014).

[15] A. B. Owen, Sobol' indices and Shapley value, SIAM/ASA Journal on Uncertainty Quantification 2 (1) (2014) 245–251.

[16] B. Efron, C. Stein, The jackknife estimate of variance, Annals of Statistics 9 (3) (1981) 586–596.

[17] D. Bigoni, A. Engsig-Karup, Uncertainty quantification with applications to engineering problems, Ph.D. thesis, Technical University of Denmark (2015).

[18] A. Saltelli, S. Tarantola, F. Campolongo, M. Ratto, Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models, Halsted Press, New York, NY, USA, 2004.

[19] A. B. Owen, Variance components and generalized Sobol' indices, SIAM Journal on Uncertainty Quantification 1 (1) (2013) 19–41.

[20] G. Beylkin, J. Garcke, M. J. Mohlenkamp, Multivariate regression and machine learning with sums of separable functions, SIAM Journal on Scientific Computing 31 (3) (2009) 1840–1857.

[21] D. Bigoni, A. Engsig-Karup, Y. Marzouk, Spectral tensor-train decomposition, SIAM Journal on Scientific Computing 38 (4) (2016) A2405–A2439.

[22] T. Homma, A. Saltelli, Importance measures in global sensitivity analysis of nonlinear models, Reliability Engineering & System Safety 52 (1) (1996) 1–17.

[23] E. Fock, Global sensitivity analysis approach for input selection and system identification purposes – a new framework for feedforward neural networks, IEEE Transactions on Neural Networks and Learning Systems 25 (8) (2014) 1484–1495.

[24] T. Abualrub, A. Jarrah, S. Kallel, H. Sulieman, mathematics Across Contemporary Sciences: AUS-ICMS, Sharjah, UAE, April 2015, Springer Proceedings in Mathematics & Statistics, Springer International Publishing, 2017.

[25] G. Hooker, Discovering additive structure in black box functions, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004, pp. 575–580.

[26] R. A. Harshman, Foundations of the PARAFAC procedure: Models and conditions for an "Explanatory" multi–modal factor analysis, UCLA Working Papers in Phonetics 16 (1970) 1–84.

[27] V. de Silva, L.-H. Lim, Tensor rank and the ill-posedness of the best low-rank approximation problem, SIAM Journal on Matrix Analysis and Applications 30 (3) (2008) 1084–1127.

[28] L. R. Tucker, Some mathematical notes on three-mode factor analysis, Psychometrika 31 (3) (1966) 279–311.

[29] L. de Lathauwer, B. de Moor, J. Vandewalle, On the best rank-1 and rank-$(R_1, R_2, ..., R_N)$ approximation of higher-order tensors, SIAM Journal of Matrix Analysis and Applications 21 (4) (2000) 1324–1342.

[30] I. V. Oseledets, E. E. Tyrtyshnikov, TT-cross approximation for multidimensional arrays, Linear Algebra Applications 432 (1) (2010) 70–88.

[31] D. V. Savostianov, I. V. Oseledets, Fast adaptive interpolation of multidimensional arrays in tensor train format, in: Proceedings International Workshop on Multidimensional (nDS) Systems, 2011, pp. 1–8. doi:https://doi.org/10.1109/nDS.2011.6076873.

[32] A. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, D. P. Mandic, Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions, Foundations and Trends in Machine Learning 9 (4-5) (2016) 249–429.

[33] N. Lee, A. Cichocki, fundamental tensor operations for large-scale data analysis using tensor network formats, Multidimensional Systems and Signal Processing (2017) 1–40.

[34] I. Oferkin, D. Zheltkov, E. Tyrtyshnikov, A. Sulimov, D. Kutov, V. Sulimov, Evaluation of the docking algorithm based on tensor train global optimization, Bulletin of the South Ural State University: Mathematical Modelling and Programming 8 (4) (2015) 83–99.

[35] A. Y. Mikhalev, I. V. Oseledets, Rectangular maximum-volume submatrices and their applications, arXiv: 1502.07838 (2015).
URL http://arxiv.org/abs/1502.07838

[36] ttpy: Python implementation of the TT-toolbox, http://github.com/oseledets/ttpy.

[37] M. Espig, W. Hackbusch, A. Litvinenko, H. G. Matthies, E. Zander, Efficient analysis of high dimensional data in tensor formats, in: Sparse Grids and Applications, Springer Berlin Heidelberg, 2013, pp. 31–56.

[38] A. Litvinenko, H. G. Matthies, T. A. El-Moselhy, Sampling and low-rank tensor approximation of the response surface, in: International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, Springer Berlin Heidelberg, 2013, pp. 535–551. doi:http://dx.doi.org/10.1007/978-3-642-41095-6_27.

[39] N. Vervliet, O. Debals, L. Sorber, L. D. Lathauwer, Breaking the curse of dimensionality using decompositions of incomplete tensors: Tensor-based scientific computing in big data analysis., IEEE Signal Processing Magazine 31 (5) (2014) 71–79.

[40] R. Ballester-Ripoll, E. G. Paredes, R. Pajarola, A surrogate visualization model using the tensor train format, in: SIGGRAPH ASIA 2016 Symposium on Visualization, 2016, pp. 13:1–13:8.

[41] Z. Zhang, X. Yang, I. V. Oseledets, G. E. Karniadakis, L. Daniel, Enabling high-dimensional hierarchical uncertainty quantification by ANOVA and tensor-train decomposition, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 34 (1) (2015) 63–76.

[42] I. V. Oseledets, D. V. Savostianov, E. E. Tyrtyshnikov, Tucker dimensionality reduction of three-dimensional arrays in linear time, SIAM Journal on Matrix Analysis and Applications 30 (3) (2008) 939–956.

[43] C. Caiafa, A. Cichocki, generalizing the column–row matrix decomposition to multi-way arrays, Linear Algebra Applications 433 (3) (2010) 557–573.

[44] S. V. Dolgov, D. V. Savostyanov, Alternating minimal energy methods for linear systems in higher dimensions, SIAM Journal on Scientific Computing 36 (5) (2014) A2248–A2271.

[45] L. Grasedyck, D. Kressner, C. Tobler, A literature survey of low-rank tensor approximation techniques, GAMM-Mitteilungen 36 (1) (2013) 53–78.

[46] I. V. Oseledets, E. E. Tyrtyshnikov, Tensor-tree decomposition does not need a tree, in: Linear Algebra and its Applications, 2010.

[47] G. Geraci, P. Congedo, R. Abgrall, G. Iaccarino, High-order statistics in global sensitivity analysis: Decomposition and model reduction, Computer Methods in Applied Mechanics and Engineering 301 (2016) 80 – 115.

[48] I. M. Sobol', Multidimensional Quadrature Formulas and Haar Functions, Nauka, Moskow (in Russian).

[49] I. Oseledets, E. Tyrtyshnikov, Algebraic wavelet transform via quantics tensor train decomposition., SIAM Journal on Scientific Computing 33 (3) (2011) 1315–1328.

[50] A. Novikov, M. Trofimov, I. Oseledets, Exponential machines, arXiv preprint.
URL `https://arxiv.org/pdf/1605.03795.pdf`

[51] J. Herman, W. Usher, SALib: An open-source python library for sensitivity analysis, The Journal of Open Source Software 2 (9).
URL `https://doi.org/10.21105/joss.00097`

[52] I. Sobol', Theorems and examples on high dimensional model representation, Reliability Engineering & System Safety 79 (2) (2003) 187–193.

[53] R. Kenett, S. Zacks, modern Industrial Statistics: Design and Control of Quality and Reliability, Duxbury Press, 1998.

[54] C. Nugteren, V. Codreanu, Cltune: A generic auto-tuner for OpenCL kernels, in: International Symposium on Embedded Multicore/Many-core Systems-on-Chip, 2015, pp. 195–202.

[55] T. L. Falch, A. C. Elster, Machine learning-based auto-tuning for enhanced performance portability of OpenCL applications, Concurrency and Computation: Practice and Experience 29 (8).